Элементы технического зрения

ТОЧЕЧНЫЕ ОСОБЕННОСТИ НА ИЗОБРАЖЕНИИ. СОВМЕЩЕНИЕ ИЗОБРАЖЕНИЙ.

Ключевые (особые) точки

- Под ключевыми точками понимаются некоторые участки картинки, которые являются отличительными для данного изображения.
- Подобные точки каждый алгоритм определяет по своему.

Составляющие процесса детектирования

- Детектор (feature detector) — осуществляет поиск ключевых точек на изображении.
- Дескриптор (descriptor extractor) — производит описание найденных ключевых точек, оценивая их позиции через описание окружающих областей.
- Матчер (matcher) — осуществляет построение соответствий между двумя наборами точек изображений.

Определение

- Особая точка m, или точечная особенность (англ. point feature, key point, feature), изображения – это точка изображения, окрестность которой o(m) можно отличить от окрестности любой другой точки изображения o(n) в некоторой другой окрестности особой точки o2(m).
- В качестве окрестности точки изображения для большинства алгоритмов берётся прямоугольное окно, составляющее размер 5x5 пикселей.

Подходы к определению

- Основанные на интенсивности изображения.
- Использующие контуры изображения: ищут места с максимальным значением кривизны или делают полигональную аппроксимацию контуров и определяют пересечения. Эти методы чувствительны к окрестностям пересечений, поскольку извлечение часто может быть неправильным в тех местах, где пересекаются 3 или более краев.
- На основе использования модели: используются модели с интенсивностью в качестве параметров, которые подстраиваются к изображениям-шаблонам до субпиксельной точности.

Классический подход – изменение яркости

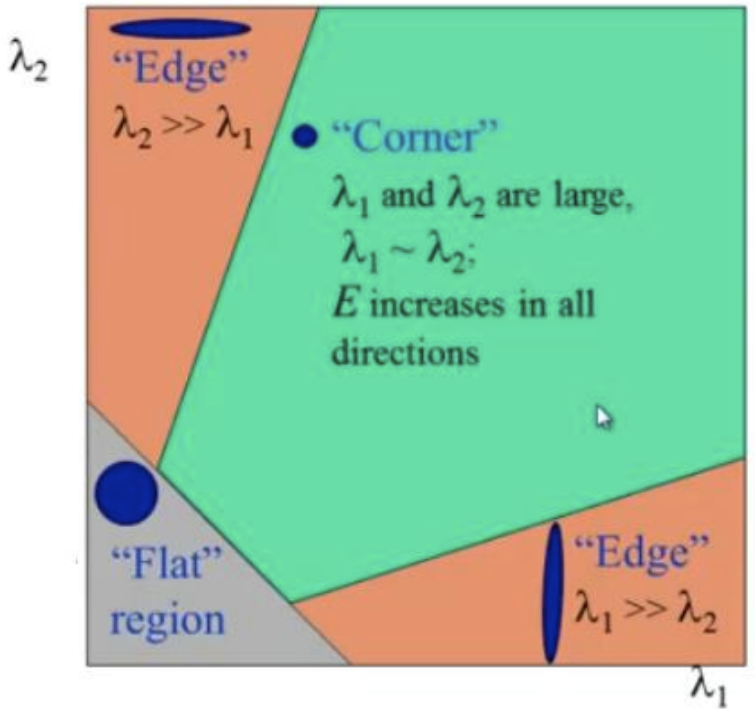
Детектор Харриса

- Вычисляем собственные значения матрицы

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}$$

- Вычисляем меру отклика

$$\mathrm{R} = \det A^T A - k \cdot tr A^T A$$



Детектор Ши-Томаси (Shi-Tomasi)

Аналогичен детектору Харриса, за исключение финального шага. В Shi-Tomasi вычисляется значение

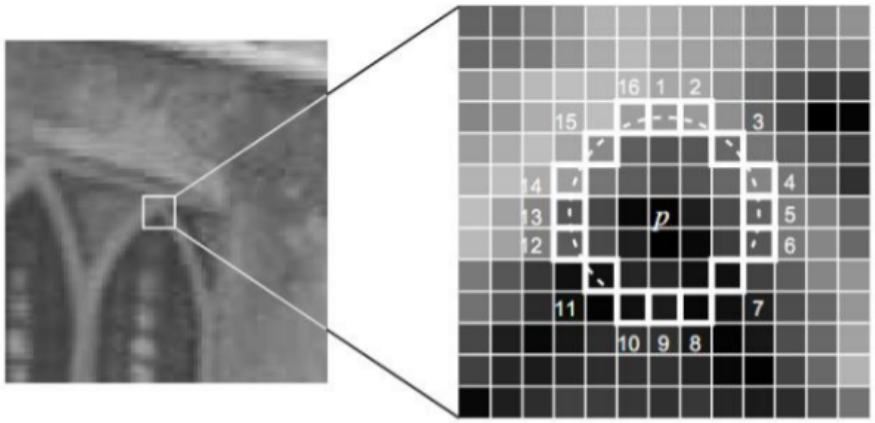
$$R = \min(\lambda_1, \lambda_2)$$

поскольку делается предположение, что поиск углов будет более стабильным.

В OpenCV реализуется в функции

cv::goodFeaturesToTrack

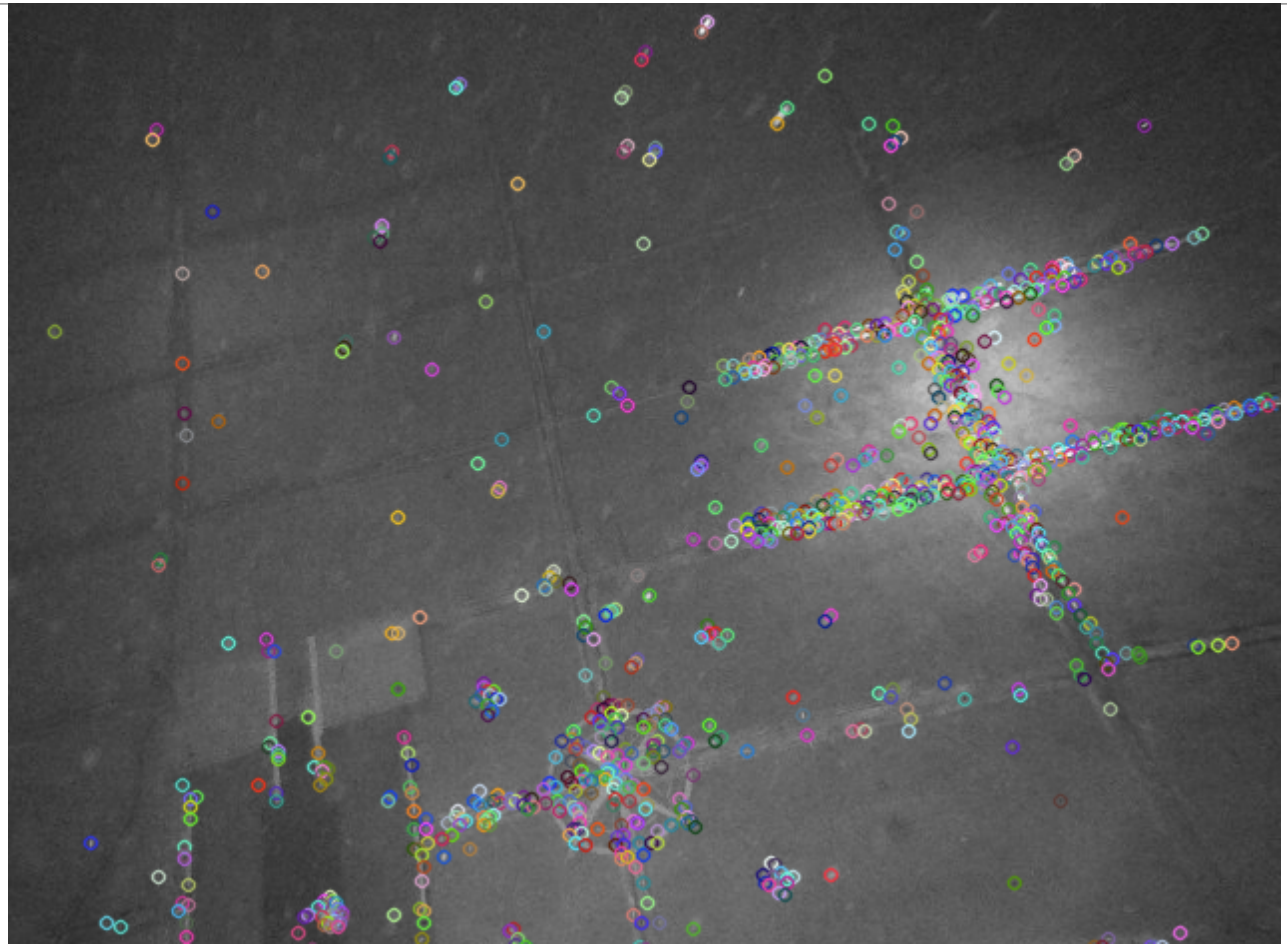
FAST



Алгоритм:

- Рассматривается окружность из 16 точек
- Точка считается особой, если на окружности есть N смежных точек, яркость которых больше $I_p + t$ или меньше $I_p - t$
- Оптимизации – проверить точки 1, 5, 9, 13
- Очень быстрый алгоритм

FAST



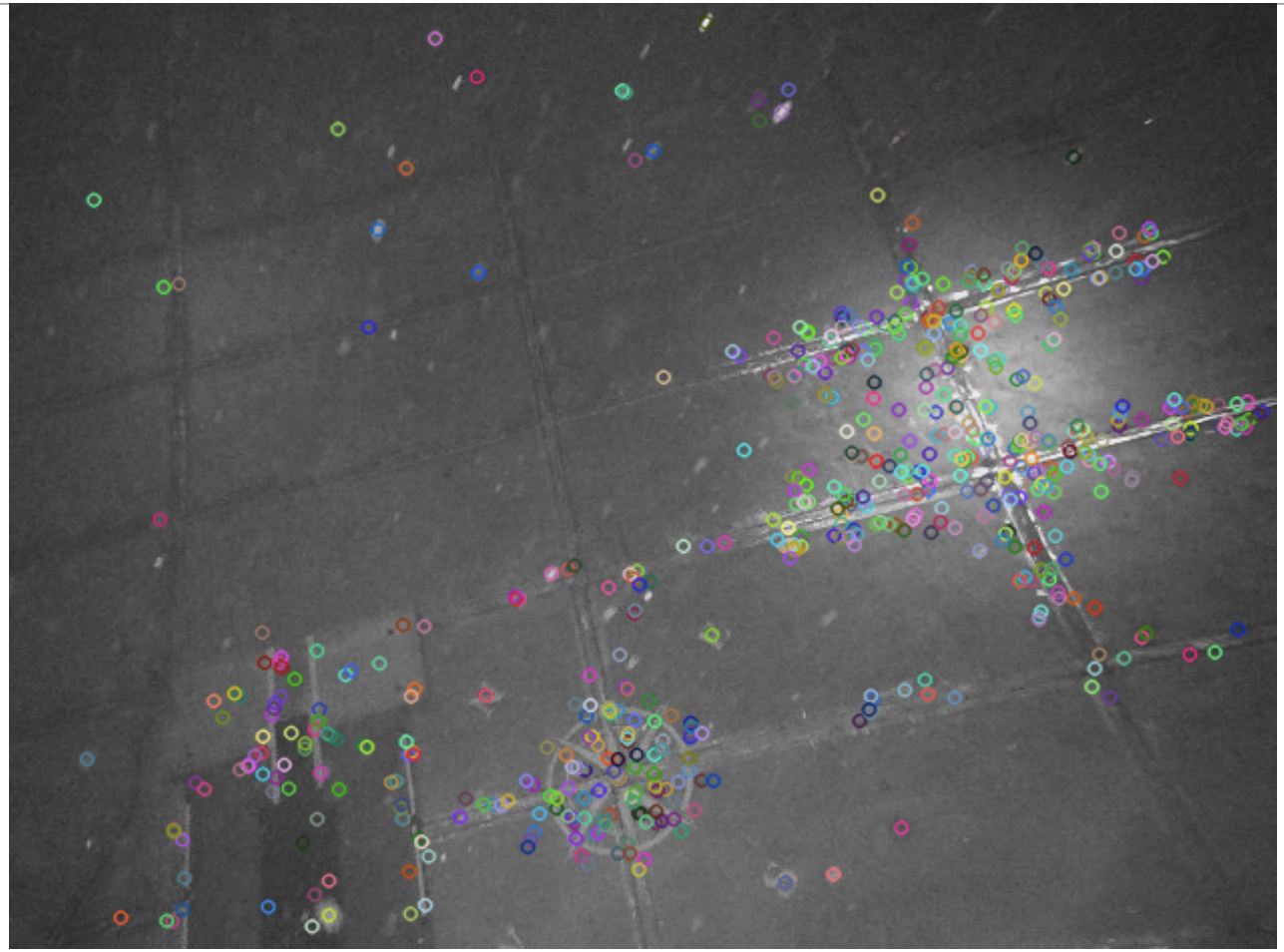
SURF

- Speeded Up Robust Features
- Решает две задачи – поиск особых точек изображения и создание их дескрипторов
- Обнаружение особых точек в SURF основано на вычислении детерминанта матрицы Гессе (гессиана)

$$H\big(f(x,y)\big) = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

$$R = \det H = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

SURF

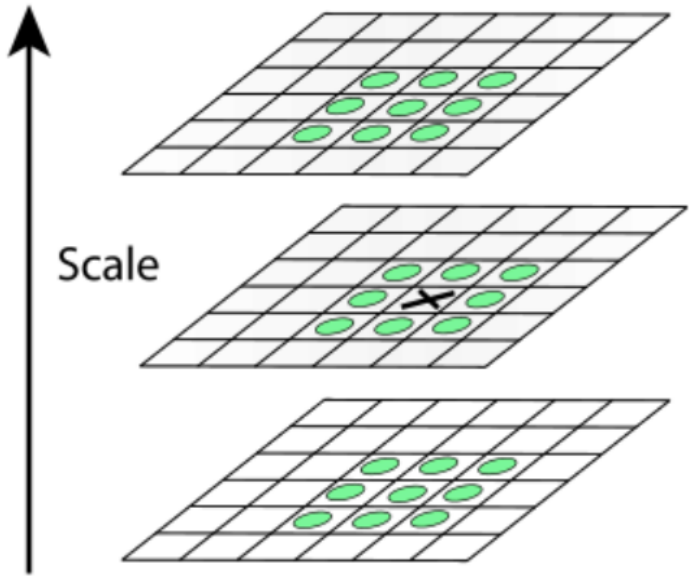


SIFT

- Основным моментом в детектировании особых точек является построение пирамиды гауссианов (Gaussian) и разностей гауссианов (Difference of Gaussian, DoG).
- Гауссианом (или изображением, размытым гауссовым фильтром) является изображение
- Разностью гауссианов называют изображение, полученное путем попиксельного вычитания одного гауссина исходного изображения из гауссиана с другим радиусом размытия.

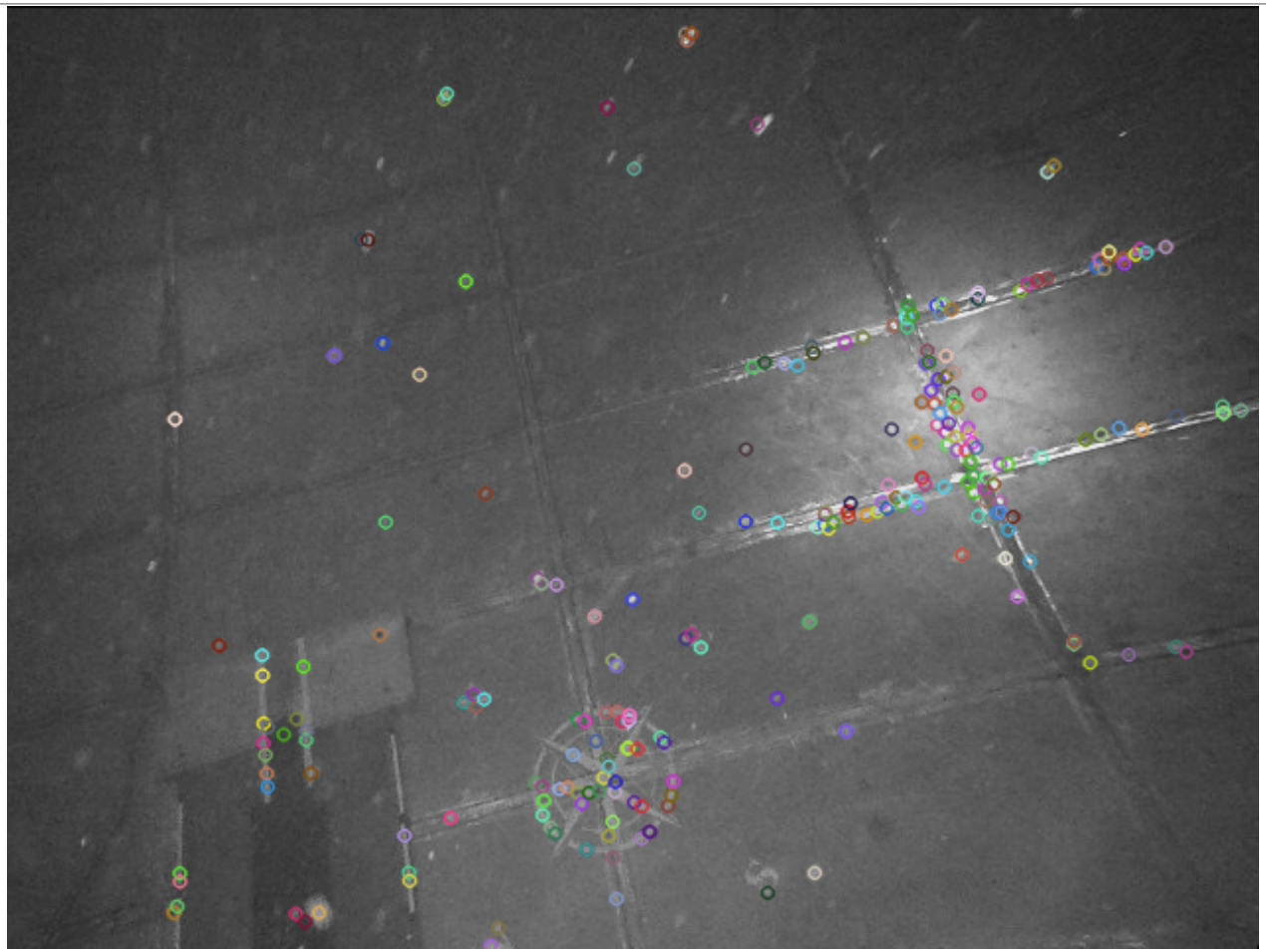
$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] \approx * I(x, y)$$

SIFT



Будем считать точку особой, если она является локальным экстремумом разности гауссианов

SIFT

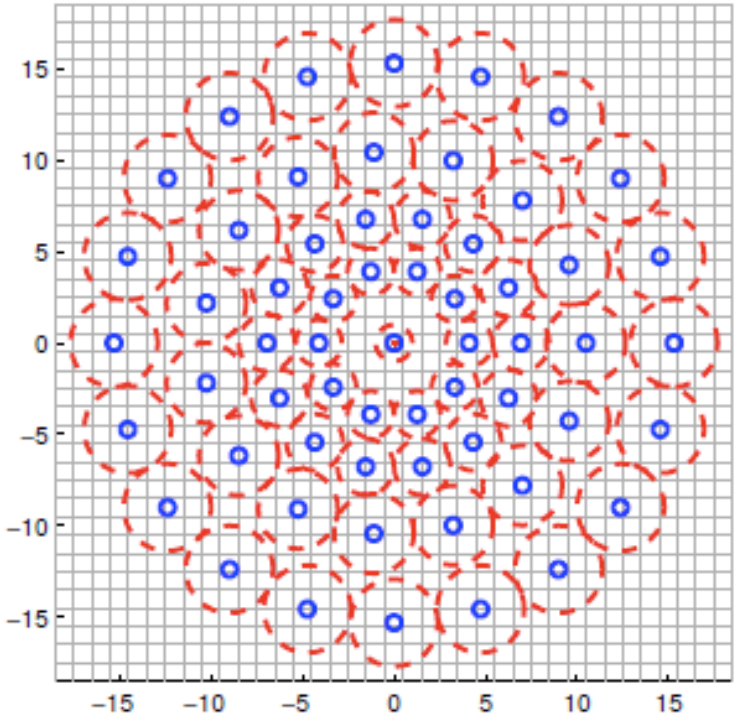


Точечный детектор BRISK

Выделение точечных особенностей - FAST

Построение их описаний

Daisy-like дескриптор



```cpp
#include <opencv2/features2d.hpp>
#include <opencv2/xfeatures2d.hpp>
cv::Mat src = cv::imread( "image.jpg", 1 );
cv::Mat imageWithKeypoints;

cv::Ptr<cv::FeatureDetector> detector;
cv::Ptr<cv::DescriptorExtractor> extractor;
cv::BFMatcher matcher;

detector = cv::FastFeatureDetector::create();
detector = cv::xfeatures2d::SURF::create();
detector = cv::xfeatures2d::SIFT::create();

std::vector<cv::KeyPoint> keys1;
detector->detect(src, keys1);

cv::drawKeypoints(src, keys1, imageWithKeypoints);
```

Загрузка видео-последовательности

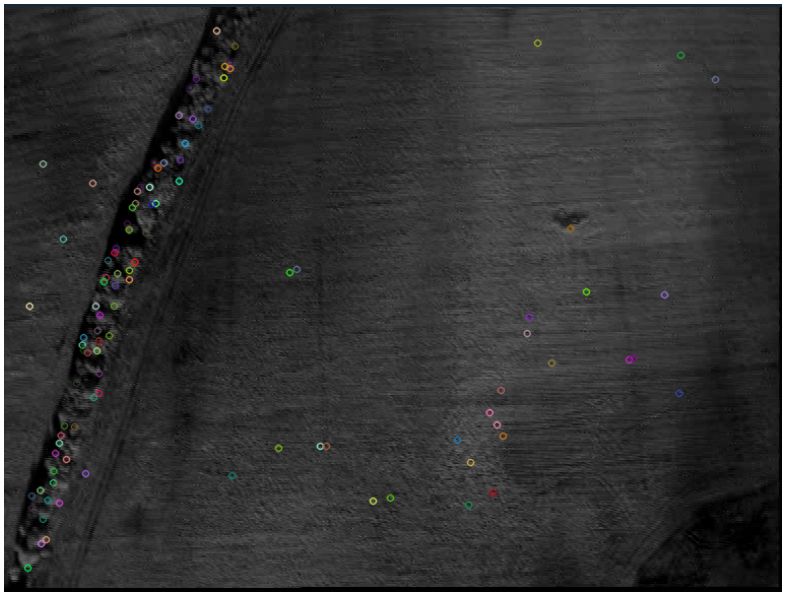
cv::VideoCapture(filename)

```
cv::Mat src;
cv::VideoCapture cap("sample_mpg.avi");
if (!cap.isOpened())
        return -1;

bool stop = false;
// Определим частоту кадров на видео
double rate = cap.get(cv::CAP_PROP_FPS);
// Рассчитаем задержку в миллисекундах
int delay = 1000 / rate;
cout << "Frame rate of video is " << rate << endl;
while (!stop)
{
        // Проверяем доступность кадра
        bool result = cap.grab();
        // Если он готов, считываем
        if (result)
                cap >> src;
        else
                continue;

        cv::imshow("Original", src);
        // Ждём нажатия на кнопку
        int key = cv::waitKey(delay);
        if (key==27) // Если это 0x27, т.е. ESC
                stop=true; // Выходим
}
```

Точечный детектор BRISK



Точечный детектор BRISK

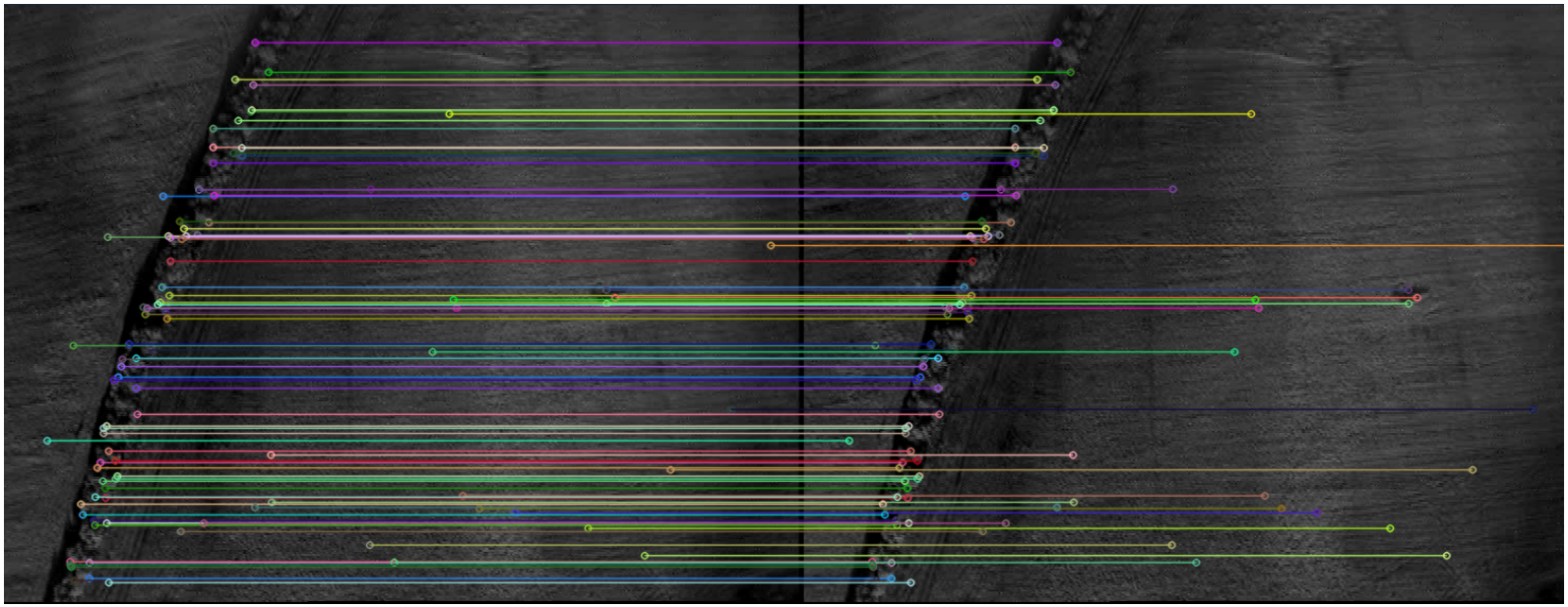
После выделения точек – можно построить их описание

По описаниям точки можно совместить

Попробуем отобразить их вывод!

```
detector = cv::BRISK::create();
extractor = cv::BRISK::create();
detector->detect(src,keys1);
detector->detect(src2,keys2);
cv::Mat descr1, descr2, img_matches;
extractor->compute(src, keys1, descr1);
extractor->compute(src2, keys2, descr2);
std::vector<cv::DMatch> matches;
matcher.match(descr1, descr2, matches);
cv::drawMatches(src, keys1, src2, keys2,\\
 matches, img_matches);
cv::imshow("matches", img_matches);
```

Точечный детектор BRISK



Что нужно подключить

ЗАГОЛОВКИ

#include<opencv2/opencv.hpp>

- Общий заголовок для OpenCV

#include<opencv2/features2d.hpp>

- Детекторы FAST, BRISK, ORB

#include <opencv2/xfeatures2d.hpp>

- Экспериментальные или лицензированные детекторы (SIFT SURF)

БИБЛИОТЕКИ

-lopencv_core

- Основная часть, cv::Mat

-lopencv_highgui

- Интерфейс, imshow()

-lopencv_features2d

- Детекторы

-lopencv_xfeatures2d

- Экспериментальные и лицензированные детекторы

-lopencv_videoio

- cv::VideoCapture

Задача 5

Используйте прилагающийся видеофайл

Напишите программу для попарного сопоставления кадров:

1. Реализуйте считывание кадров в фреймы
2. Найдите ключевые точки и постройте описания
3. Выведите результат сопоставления на экран
4. Используйте дескрипторы SURF, SIFT и BRISK